HW2 Dry

1.

```kotlin
class Observable<A> private constructor(private val value: A) {
   private var observers = mutableListOf<(Any) -> Unit>()

   companion object {
       fun <A> of(a: A) = Observable(a)
   }

   fun <B> flatMap(f: (A) -> Observable<B>): Observable<B> {
       val newVal = f(value)
       observers.forEach { it(newVal.value as Any) }
       newVal.observers = observers
       return newVal

   }

   fun subscribe(f: (Any) -> Unit) = observers.add(f)
   fun unsubscribe(f: (Any) -> Unit) = observers.remove(f)
}
```

2.
Left Identity:
`Observable.of(value).flatMap(f) == f(value)`

Right identity:
`m.flatMap{ Observable.of(it) } == m`
Value in m is the same, and the observers list is also copied so it is equivalent monad.

Associativity
`m.flatMap(f).flatMap(g) === m.flatMap{ f(it).flatMap(g) }`
Value is obviously the same at both sides, as it transferred simply by invoking f as in the simple Box example. The list of observers in lhs is equal to rhs, since we never removing and we always propagate it via flatMap.